Here's a **step-by-step beginner-friendly guide** to create a **Django REST API** for a table with:

`id, name, city, phone, email`

We'll use:

- Django
- Django REST Framework

□ STEP 1: Install Django + DRF

Create virtual environment in your windows pc

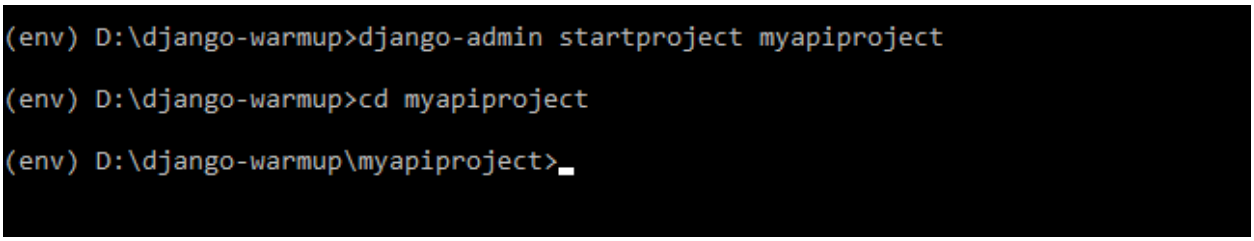
```
python -m venv env

env\Scripts\activate
```

Install packages

```
pip install django djangorestframework
```

□ STEP 2: Create Project

```
django-admin startproject myapiproject
cd myapiproject
```



□ STEP 3: Create App

```
python manage.py startapp api
```

```
(env) D:\django-warmup\myapiproject>python manage.py startapp api
```

□ STEP 4: Add Apps in settings.py

Open `myproject/settings.py`

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

    'rest_framework',  # DRF
    'api',             # your app
]
```

□ STEP 5: Create Model (Database Table)

In `api/models.py`

```
from django.db import models

class Student(models.Model):
    name = models.CharField(max_length=100)
    city = models.CharField(max_length=100)
    phone = models.CharField(max_length=15)
    email = models.EmailField()

    def __str__(self):
        return self.name
```

□ STEP 6: Run Migrations

```
python manage.py makemigrations
python manage.py migrate
```

```
(env) D:\django-warmup\myapiproject>python manage.py makemigrations
Migrations for 'api':
  api\migrations\0001_initial.py
    + Create model Student

(env) D:\django-warmup\myapiproject>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, api, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
```

□ STEP 7: Create Serializer (VERY IMPORTANT)

Create file:

□ api/serializers.py

```
from rest_framework import serializers
from .models import Student

class StudentSerializer(serializers.ModelSerializer):
    class Meta:
        model = Student
        fields = '__all__'
```

□ This converts database ↔ JSON

STEP 8: ViewSet (CORE CONCEPT)

Instead of writing GET/POST/PUT/DELETE manually, we use:

 `ModelViewSet`

 `api/views.py`

```
from rest_framework import viewsets
from .models import Student
from .serializers import StudentSerializer

class StudentViewSet(viewsets.ModelViewSet):
    queryset = Student.objects.all()
    serializer_class = StudentSerializer
```

WHAT THIS AUTOMATICALLY GIVES YOU

Without writing extra code, you now get:

Action	HTTP Method	URL
List all students	GET	/students/
Create student	POST	/students/
Retrieve student	GET	/students/1/
Update student	PUT	/students/1/
Delete student	DELETE	/students/1/

□ STEP 9: ROUTER (AUTO URL GENERATION)

□ api/urls.py

```
from django.urls import path, include
from rest_framework.routers import DefaultRouter
from .views import StudentViewSet

router = DefaultRouter()
router.register('students', StudentViewSet, basename='student')

urlpatterns = [
    path('', include(router.urls)),
]
```

□ STEP 5: MAIN PROJECT URL

□ myproject/urls.py

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/', include('api.urls')),
]
```

□ STEP 6: RUN SERVER

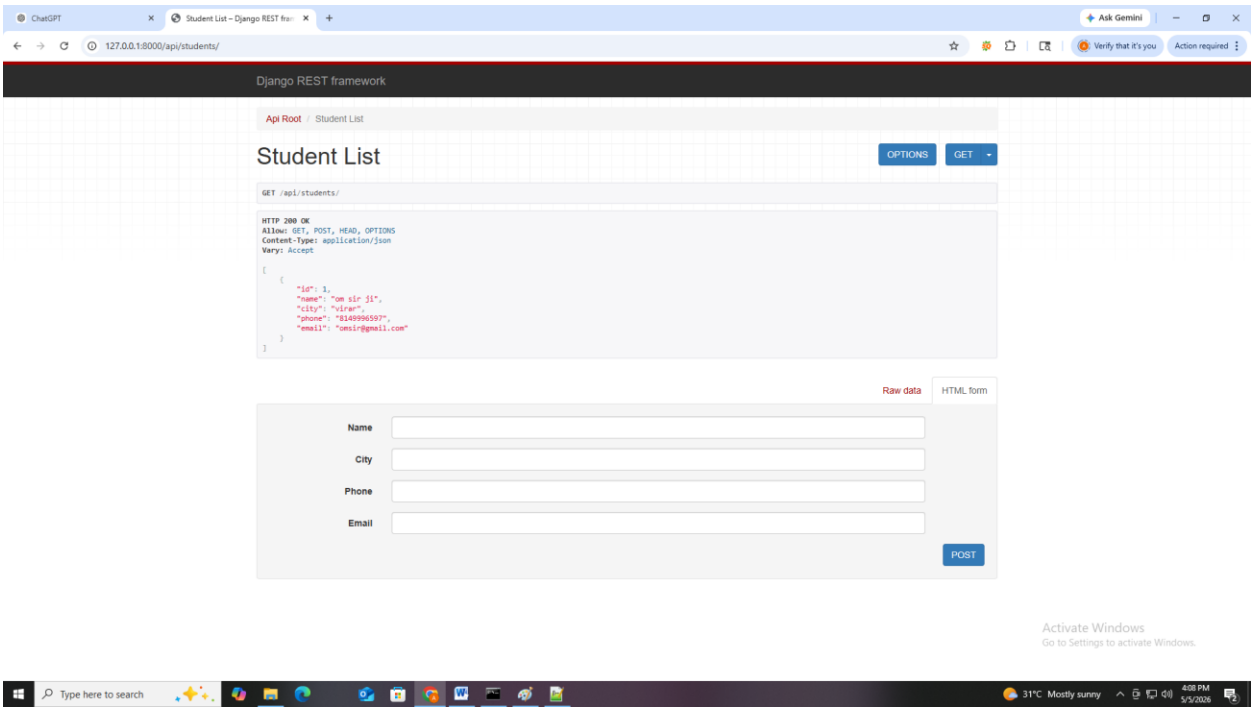
```
python manage.py runserver
```

□ STEP 7: TEST API

Now open browser/Postman:

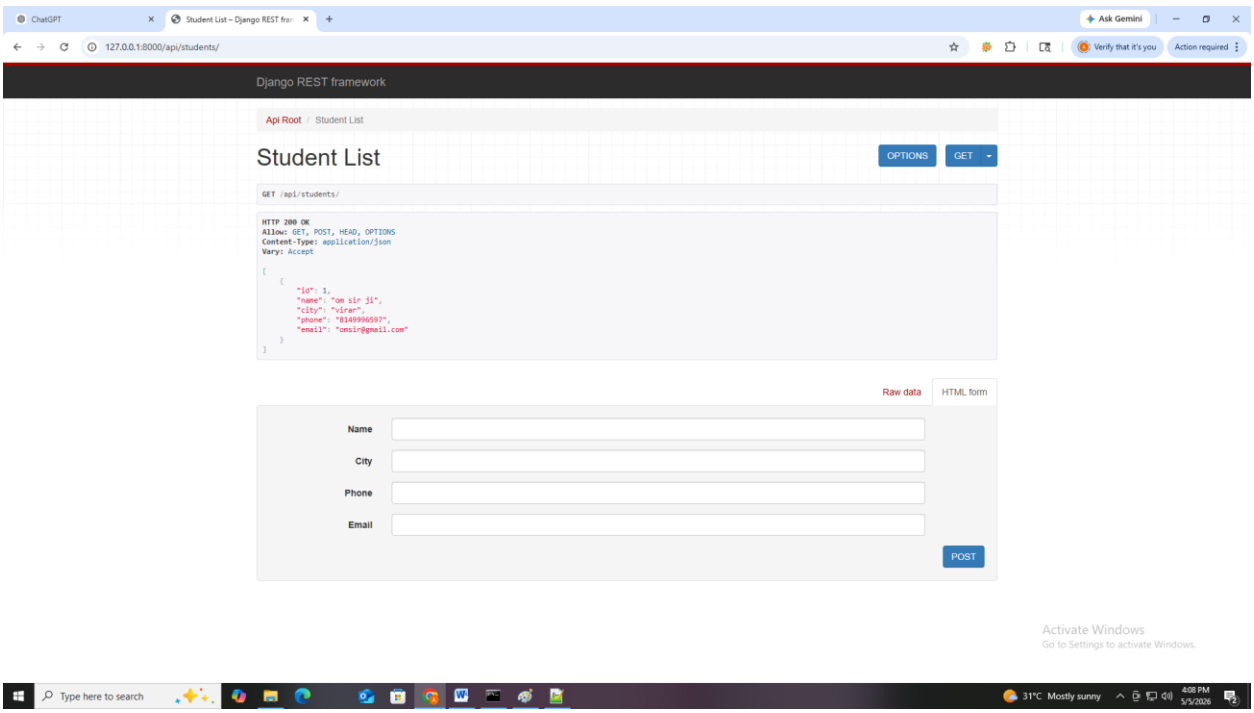
List Students GET :-

http://127.0.0.1:8000/api/students/



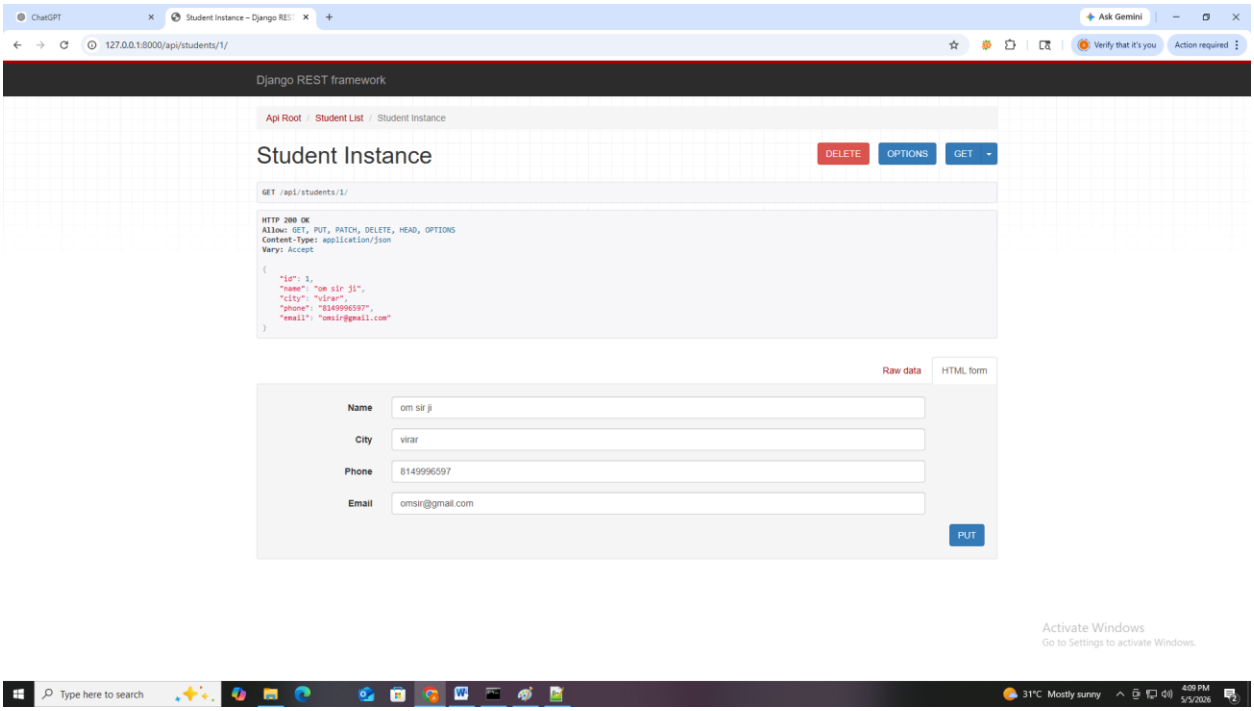
Create Student

```
POST http://127.0.0.1:8000/api/students/
{
  "name": "Rahul",
  "city": "Mumbai",
  "phone": "9876543210",
  "email": "rahul@example.com"
}
```



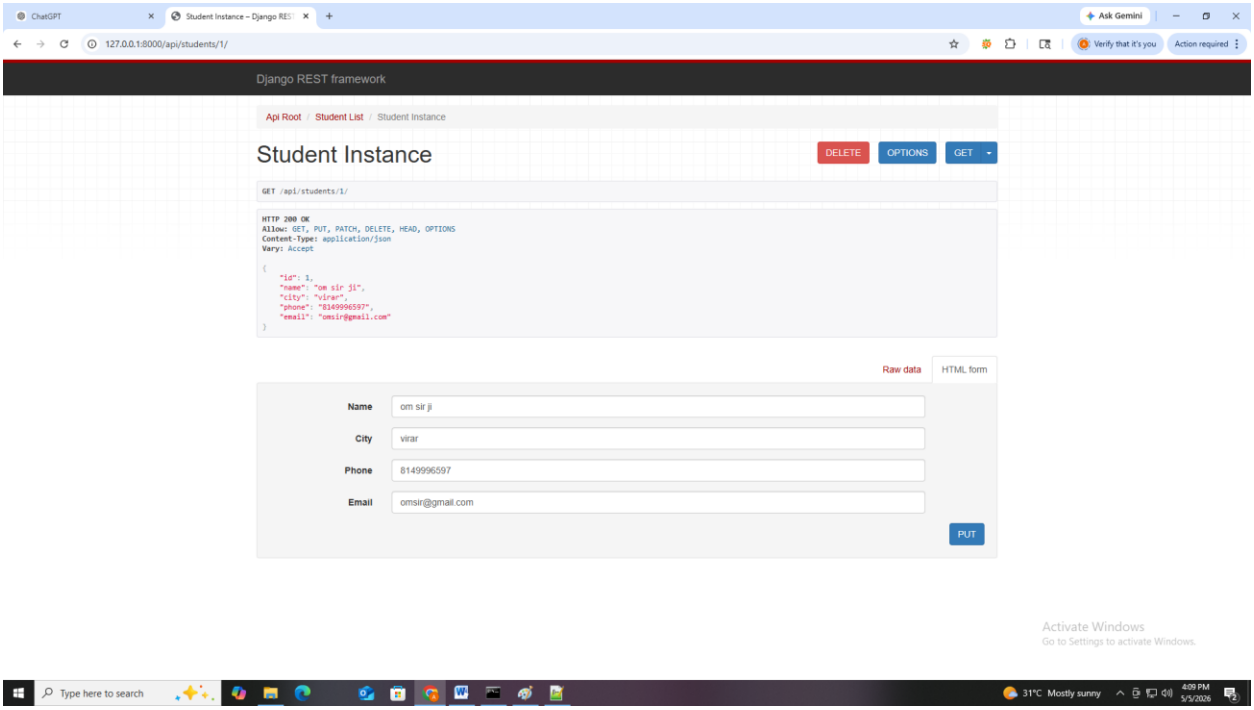
Get Single Student GET :-

http://127.0.0.1:8000/api/students/1/



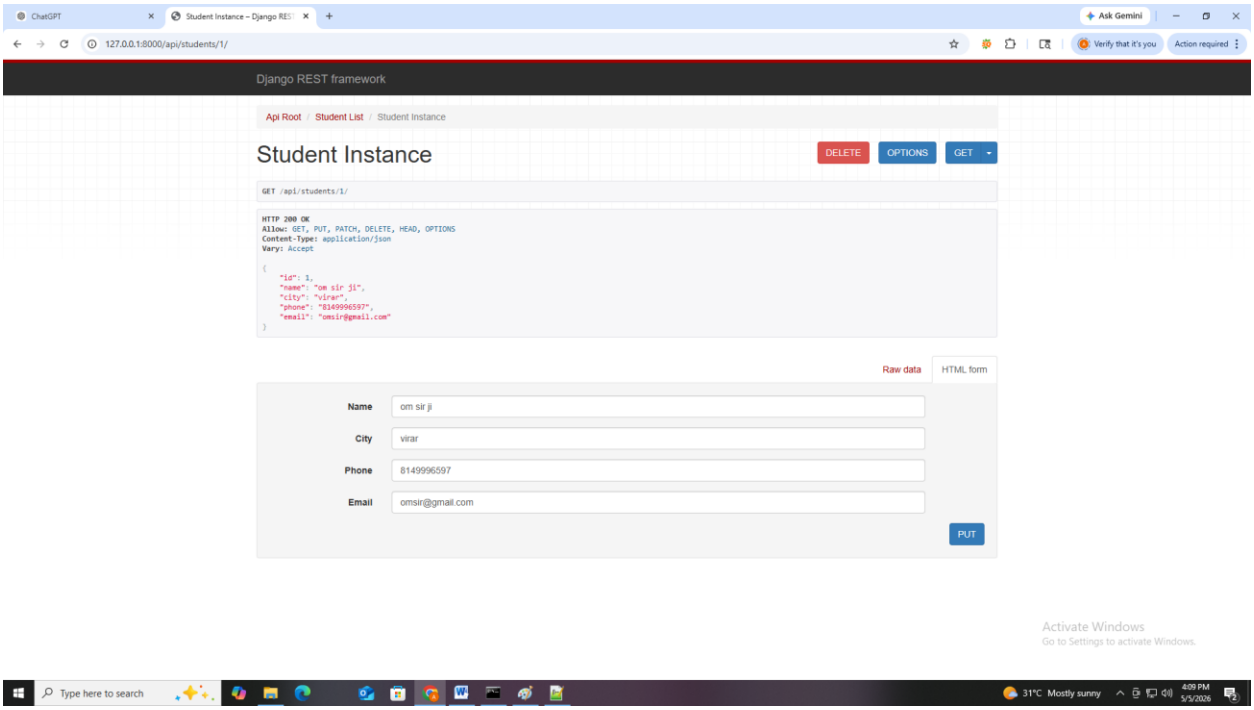
Update Student PUT :-

http://127.0.0.1:8000/api/students/1/



Delete Student

```
DELETE /api/students/1/
```



1. serializers.py (Meaning of each line)

Code:

```
from rest_framework import serializers
from .models import Student

class StudentSerializer(serializers.ModelSerializer):
    class Meta:
        model = Student
        fields = '__all__'
```

Line-by-line meaning

□ Line 1

```
from rest_framework import serializers
```

□ Imports Django REST Framework's serializer module.
✔ Serializers convert:

- Python objects ↔ JSON (API format)

□ Line 2

```
from .models import Student
```

□ Imports your database table (model) named `Student` from the same app.

□ Line 3

```
class StudentSerializer(serializers.ModelSerializer):
```

□ Creates a serializer class.

✔ `ModelSerializer` means:

- Django automatically generates fields from model
- No need to manually define each field

□ Line 4–7

```
class Meta:
    model = Student
    fields = '__all__'
```

□ Meaning:

✔class Meta

□ Special inner class that gives configuration to serializer

✔model = Student

□ Tells serializer:

“Use Student model (database table)”

✔fields = '__all__'

□ Include ALL fields:

- id
- name
- city
- phone
- email

✔ Alternative:

```
fields = ['name', 'email']
```

□ 2. views.py (ModelViewSet)

Code:

```
from rest_framework import viewsets
from .models import Student
from .serializers import StudentSerializer

class StudentViewSet(viewsets.ModelViewSet):
    queryset = Student.objects.all()
    serializer_class = StudentSerializer
```

Line-by-line meaning

□ Line 1

```
from rest_framework import viewsets
```

□ Imports ViewSet system from DRF

✔ ViewSet = combines all CRUD operations:

- GET
- POST
- PUT
- DELETE

□ Line 2

```
from .models import Student
```

□ Imports database table (Student model)

□ Line 3

```
from .serializers import StudentSerializer
```

□ Imports serializer to convert:

- Model ↔ JSON

□ Line 4

```
class StudentViewSet(viewsets.ModelViewSet):
```

□ Creates a ViewSet class

✔ `ModelViewSet` means:

Django automatically gives full CRUD API without writing methods

---

□ Line 5

```
queryset = Student.objects.all()
```

□ Meaning:

- Fetch ALL students from database
- Equivalent to SQL:

```
SELECT * FROM student;
```

□ Line 6

```
serializer_class = StudentSerializer
```

□ Tells ViewSet:

"Use this serializer to convert data to/from JSON"

What this ViewSet automatically gives you:

Action	Method	URL
List	GET	/students/
Create	POST	/students/
Retrieve	GET	/students/1/

Action	Method	URL
Update	PUT	/students/1/
Delete	DELETE	/students/1/

□ 3. urls.py (Router system)

Code:

```
from django.urls import path, include
from rest_framework.routers import DefaultRouter
from .views import StudentViewSet

router = DefaultRouter()
router.register('students', StudentViewSet, basename='student')

urlpatterns = [
    path('', include(router.urls)),
]
```

Line-by-line meaning

□ Line 1

```
from django.urls import path, include
```

□ Imports URL tools:

- `path()` → define routes
- `include()` → connect other URL files

□ Line 2

```
from rest_framework.routers import DefaultRouter
```

 Imports Router system from DRF

✔ Router automatically generates URLs for ViewSets

Line 3

```
from .views import StudentViewSet
```

 Imports your ViewSet (API logic)

Line 4

```
router = DefaultRouter()
```

 Creates a router object

✔ Think of it as:

“Auto URL generator”

Line 5

```
router.register('students', StudentViewSet, basename='student')
```

Meaning:

- `'students'` → URL name
- `StudentViewSet` → logic to handle requests
- `basename='student'` → name used internally for URL naming

✔ This creates URLs like:

- `/students/`
- `/students/1/`

Line 6–8

```
urlpatterns = [
    path('', include(router.urls)),
]
```

Meaning:

- `router.urls` contains all auto-generated URLs
- `include()` attaches them to project routing

FINAL SIMPLE SUMMARY

serializer.py

Converts data:

Database ↔ JSON